

Automated Discovery of Search-Extension Features

Pálmi Skowronski¹, Yngvi Björnsson¹, and Mark H.M. Winands²

¹ Reykjavík University, School of Computer Science,
Menntavegi 1, Reykjavík 101, Iceland
{palmis01,yngvi}@ru.is

² Games and AI Group, Department of Knowledge Engineering,
Maastricht University, Maastricht, The Netherlands
m.winands@maastrichtuniversity.nl

Abstract. One of the main challenges with selective search extensions is designing effective move categories (features). Usually, it is a manual trial-and-error task, which requires both intuition and expert human knowledge. Automating this task potentially enables the discovery of both more complex and more effective move categories. The current work introduces *Gradual Focus*, an algorithm for automatically discovering interesting move categories for selective search extensions. The algorithm iteratively creates new more refined move categories by combining features from an atomic feature set. Empirical data is presented for the game Breakthrough showing that Gradual Focus looks at a number of combinations that is two orders of magnitude fewer than a brute-force method does, while preserving adequate precision and recall.

1 Introduction

The $\alpha\beta$ algorithm is one of the fundamental and most effective search techniques used by game-playing programs for playing two-person adversary board games, such as chess and checkers. Over the years many enhancements have been proposed to improve its efficiency. For instance, we know that the standard strategy of exploring all alternatives to the same fixed depth is not most effective. Instead, various techniques have been proposed for searching the game tree more selectively, where some lines of play are terminated prematurely whereas others are explored more deeply. The former scenario is referred to as search reductions (or speculative pruning) and the latter as search extensions. In chess, for example, it is common to resolve forced situations, such as checks and recaptures, by searching them more deeply.

The move-decision quality of the alpha-beta algorithm is greatly influenced by the choices of which lines are investigated deeply [1,2]. Therefore, the design of an effective search-extension scheme is fundamental to any high-performance $\alpha\beta$ -based game-playing program. The typical approach for incorporating search extensions into a game-playing program is to predefine a set of move categories (e.g., checks and recaptures), and then associate a different cost weight to each

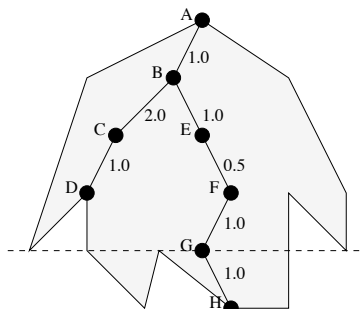


Fig. 1. Fractional-ply example

category. During the search, each move is categorized as belonging to one of the predefined move classes, and the depth of the current search path then becomes the sum of the weights of the moves on the path. If all move categories have the same weight, one would obtain the regular behaviour of a fixed-depth search. However, by assigning a weight of less than one to selected move categories, e.g., checking moves, such lines of play will be explored more deeply. This scheme is commonly referred to as fractional-ply extensions [3,4], depicted in Figure 1.

The weight of each move category, i.e., its fractional-ply value, is either manually assigned a value based on trial and error, or, alternatively, automatically tuned from game records [5,6], test-suites [7], or during play [7,8,9]. For creating the move categories the standard practice is to do it manually based on intuition and domain expertise. In this work we investigate ways for automatically discovering useful move categories for use in game-playing programs. The main contribution is a new method for automatically discovering such features, called *Gradual Focus*. We experiment with it in the game Breakthrough, where there exists little knowledge of what comprises good moves to extend on.

The paper is structured as follows. In Section 2 we give an overview of relevant background material, followed by a description of the new Gradual Focus feature-discovery algorithm in Section 3. The algorithm is empirically evaluated in Section 4, and finally we conclude and discuss future work in Section 5.

2 Background

The general approach to automated feature discovery is to start with a set of so-called *atomic* features. The atomic features are typically simple features expressing trivial facts about the problem domain, for example, type and placement of pieces. As a standalone, these features are not necessary effective, for example, because they might be too general. More sophisticated features are then constructed by combining the atomic features in various ways, e.g., by using the logical operators $\wedge$ and $\vee$. This may be done in an iterative fashion, that is, first pair-wise combinations are created, then three-wise, etc. The problem, though, is that the number of possible feature combinations grows exponentially in each